REMARKS

Claims 1, 4-8, 10-12, 14-19, and 21 are pending in the application. Claims 1, 4-8, 10-12, and 14-19 stand rejected. Claim 9 has been previously canceled. Claims 2, 3, 13, and 20 are canceled herein. Applicant requests reconsideration of the application in view of the following remarks.

Claim Objections

The examiner has objected to claims 2-4 as not being further limiting. Claims 2-3 have been canceled herein, so the Examiner's objections are moot with respect to claims 2 and 3. Claim 4, however, recites that an "implementation is provided for each computer programming interface," whereas claim one only requires an interface only for "at least one" interface. During the interview, the Examiner agreed that the objection should be withdrawn.

Claim Rejection - 35 U.S.C. §112, second paragraph

The Examiner has objected to the terminology "self-describing" interface in the claims as being unclear. The definition of "self-describing" interface is provided in the specification:

p. 5:

> As another enhancement to the present invention, common interfaces can be defined that to provide descriptive information about each trait and trait implementation. Modern programming languages, such as C++ and Java, permit the creation of an abstract class or interface for this purpose. Each trait and trait implementation can then inherit from the abstract class or support the interface. The descriptive information provided by such a common interface can include, but is not limited to, the interface required by the trait or supported by the trait implementation; the name, brief summary, description, and other documentation-related information for the trait or trait implementation; a list of the subtraits for each trait implementation; the interface required for each subtrait; numerical or descriptive measures of the expected relevance, importance, or usefulness of each trait implementation; and statistical information such as the number of constructed or active solutions containing the trait implementation, or the average fitness measurement of solutions containing the trait implementation.
>
> Traits that support such a common interface are called self-describing traits, and the methods or routines that are required by

the common interface are called self-describing methods. Similarly, trait implementations that support such a common interface are called self-describing trait implementations. Self-describing traits and trait implementations are useful in graphical user interfaces that facilitate construction of computer programs or present the modules in existing computer programs, since the self-describing methods can be used to obtain descriptions, feedback, and other information about the components required or utilized by specific selections. Self-describing traits are also useful for computer tools such as optimization algorithms and computerized solution builders, because the self-describing methods facilitate the interpretation and selection of traits within existing or newly constructed solutions.

p. 39:

Representation of Self-Describing Traits

...

Traits can be made self-describing by extending the Trait class with methods that provide descriptive information about each trait. FIG. 7 shows the information that each Trait object contains. Each complex trait, such as the TRADING_SYSTEM trait 701, is associated with a Trait object that records the programming interface for the trait and the list of possible implementations for the trait. Each trait that is simply a character string, such as the SECURITY trait 702, is associated with a Trait object that records the possible values, such as "IBM", "SUNW", and "MSFT." Each trait that is simply a number, such as the TIME_FRAME trait 703, is associated with a Trait object that records the possible range of values.

[code redacted]

Using the self-describing methods in this new definition of Trait, the surrounding code can determine – for any particular trait – what interface that trait has and what implementations are available.

...

Each trait implementation must also be made self-describing by adding a TraitImplementation interface. The information that will now be associated with each trait implementation is shown in FIG. 8, where EntryExitTradingSystem trait implementation 801 is depicted as an example.

p.43

Additional Self-Describing Methods to Support Interactive Development Environments

> The information associated with each trait and trait implementation can be extended with a name, brief description, and longer description. This information is then available for display in an interactive development environment to assist the human user in assembling useful computer programs.
>
> FIG. 9 illustrates how the information associated with a trait or trait implementation can be extended with this type of information. The information block 901 depicts the information that is now associated with the EntryExitTradingSystem trait implementation. Similarly, the information block 902 depicts the information that is now associated with the ENTRY trait.

Basically, a self-describing trait is a trait that supports an API providing descriptive information about the trait and trait implementation. Figure 7 and the associated text from the specification describes an example of a self-describing trait. As such, Applicant believes that the terminology employed in the claims and that the Examiner's rejection should be withdrawn.

Claim Rejections - 35 U.S.C. §101

The Examiner has rejected Claims 1-20 as being directed to non-statutory subject matter. Applicant respectfully traverses the rejection.

The Federal Circuit's recent *In re Bilski* decision has supposedly clarified the bounds of patent-eligible subject matter for process claims. *See In re Bilski*, 545 F.3d 943 (Fed. Cir. 2008) (en banc). The *Bilski* court held that "the machine-or-transformation test, properly applied, is the governing test for determining patent eligibility of a process under § 101." *Id.* at 956. The *Bilski* court further held that "the 'useful, concrete and tangible result' inquiry is inadequate [to determine whether a claim is patent-eligible under § 101.]" *Id.* at 959-60.

The *Bilski* court stated the machine-or-transformation test as follows: "A claimed process is surely patent-eligible under § 101 if: (1) it is tied to a particular machine or apparatus, or (2) it transforms a particular article into a different state or thing." *Id.* at 954; see also *In re Comiskey*, 499 F.3d 1365, 1377 (*Fed. Cir.* 2007) (discussing the same test from *Diehr*, 450 U.S. 175).

Process claims directed to fundamental principles – including laws of nature, natural phenomena, and abstract ideas – mental processes, or mathematical algorithms are unpatentable. *Bilski*, at 951-52. A process claim that is tied to a specific machine may be patentable under § 101. *Id.* at 961; *Comiskey*, 499 F.3d at 1377.

While the *Bilski* court did not elaborate on the "machine" branch of the test, it did provide some guidance on the issue. The court explained that "the use of a specific machine or transformation of an article must impose meaningful limits on the claim's scope to impart patent-eligibility" and "the involvement of the machine or transformation in the claimed process must not merely be insignificant extra-solution activity." *Bilski*, at 961-62 (internal citations omitted).

Turning to the "transformation" branch of the "machine-or transformation" test, claims reciting incidental transformations or extrasolution activity also do not convert an otherwise ineligible claim into an eligible one. However in *Benson*, the Supreme Court concluded that "[t]ransformation and reduction of an article 'to a different state or thing' is the clue to the patentability of a process claim that does not include particular machines." *Benson*, 409 U.S. at 70. The Court explained that several cases – *Corning v. Burden*, 15 How. (56 U.S.) 252 (1854) (tanning and dyeing), *Cochrane*, 94 U.S. 780 (manufacturing flour), *Tilghman v. Proctor*, 102 U.S. 707 (1880) (manufacturing fat acids), and *Expanded Metal Co. v. Bradford*, 214 U.S. 366 (1909) (expanding metal) – could all fairly be read to involve physical transformation of some article or material to a different state or thing. *Benson*, 409 U.S. at 69-70. *See also Bilski*, at 962-63 (discussing physical transformation and reviewing Supreme Court precedents including *Diehr* (process of curing rubber)).

Where the claims do not involve a physical transformation, the *Bilski* court explained, that the central question is "whether Applicants' claim recites a fundamental principle and, if so, whether it would pre-empt substantially all uses of that fundamental principle if allowed." *Bilski* at 954. In other words, the court distinguished between "claims that 'seek to pre-empt the use of' a fundamental principle, on the one hand, and claims that seek only to foreclose others from using a particular 'application' of that fundamental principle, on the other. *Id.* at 953 (quoting *Diehr*, 450 U.S. at 187).

The claims, as presently amended, satisfy the second prong of the Bilski test since they transform a tangible article into a different state. Moreover, the claims do not recite a fundamental principle that would pre-empt substantially all uses of that fundamental principle if

allowed. In addition, the claims are not directed to fundamental principles – including laws of nature, natural phenomena, and abstract ideas – mental processes, or mathematical algorithms.

Specifically the claims are directed to a method of constructing a computer program for solving a problem for a user. The computer program is a tangible thing whose state has been changed from a defined set of traits in which each trait characterizes a portion of a solution algorithm to the problem, a defined programming interface for at least one each of the traits, at least two implementations for at least one of the defined programming interfaces, and a specified subtrait associated with at least one of the traits or the implementations. Out of these components a computer program is constructed in a specific manner as described in claim 1 such that a first implementation that solves the user's problem is construction along with an alternate implementation that better solves the problem via an optimization routine.

Because the claims achieve the tangible result by forming a computer program, Applicant believes that the claims are patentable according to the Bilski decision and respectfully requests withdrawal of the Section 101 rejection.

Claim Rejections - 35 U.S.C. §103 (a)

Claims 1-20 stand rejected as obvious in view of U.S. Pat. Publication No. 2001/0013027 ("Akkiraju"). Applicant respectfully traverses the rejection.
Claim 1 now recites "defining a set of traits in which each trait characterizes a portion of a solution algorithm to the problem," "specifying a subtrait associated with at least one of the traits or the implementations," and "selecting a top-level trait that characterizes a solution to the problem, " These features are neither taught nor fairly suggested by Akkiraju.

The examiner takes the position that "the defining of a 'programming interface' is inherently defined for the decompositions taught by Akkiraju." Akkiraju, however, decomposing the "problem" into "subproblems". This is fundamentally different from the trait hierarchy, which is a decomposition of a **solution** – a particular computer program for solving the problem. Applicant acknowledges that both decompositions are hierarchical, but – as disclosed in the "Background of the Invention" section – decomposition into a hierarchy is, in itself, an old and well-known approach. What is novel and unobvious about the present invention as claimed is the particular approach outlined in [0042] through [0044] and elsewhere for describing a set of possible programs as a set of traits that can be composed into programs.

The Examiner has found that "in not distinctly claiming what the 'implementation' includes, that the 'implementation' could be nothing more than providing the code for the 'defined programming interfaces', which would have been obvious..." In the present invention, Applicant is claiming exactly the unique combination of elements recited in claim 1, namely, "defining a set of traits in which each trait characterizes a portion of a solution algorithm to the problem," "specifying a subtrait associated with at least one of the traits or the implementations," and "selecting a top-level trait that characterizes a solution to the problem" which are not taught or fairly suggested by Akkiraju.

Akkiraju teaches "A computer implemented optimization method" comprising the steps of "decomposing an optimization problem into a first plurality of sub-problems." This specifically refers to decomposing a **problem**. It does not, however, teach the use of any hierarchical decomposition of anything on a computer as suggested by the Examiner.

Akkiraju teaches "predicting an evaluating of a solution to a sub-problem." In contrast the present invention does not refer to either "predicting" or "sub-problems." Rather, the present invention is directed to finding a solution to the problem defined by traits and subtraits which are defined terms in the specification and which are more narrowly defined than the definitions used by the Examiner. Akkiraju does not disclose traits and subtraits as properly defined in the specification.\

Lastly Akkiraju teaches "evaluating a composed solution of said optimization problem based on at least one of an evaluation of a solution and a predicted evaluation for each sub-problem, and repeating said decomposing step to form a second plurality of sub-problems wherein at least one of said sub-problems of said second plurality of sub-problems differs from sub-problems of said first plurality of sub-problems." In contrast, the present invention does not refer to decomposing or composing problems, nor does it refer to sub-problems. Moroever, Akkiraju does not disclose selecting "solutions to the problem." Rather Akkiraju employs a subproblem decomposition scheme which is not the equivalent of a solution.

Because claims 4-8, 10-12, 14-19, and 21 depend from claim 1, and are allowable based at least upon their dependency upon an allowable independent claim. Applicant

respectfully requests that the rejection of claims 4-8, 10-12, 14-19, and 21 be withdrawn, and that claims 4-8, 10-12, 14-19, and 21 be allowed.

The remaining references cited (but not applied) have been reviewed These references are not deemed to be material to the patentability of the claimed invention. For the foregoing reasons, the claims as they now stand are patentable over the art of record, and withdrawal of the rejections and allowance of all pending claims is earnestly solicited.

****

It is respectfully submitted that the claims in the application are allowable. Reconsideration and withdrawal of all rejections are respectfully requested. Favorable notice to this effect and early Notice of Allowance are earnestly solicited.

Should the examiner have any questions and in order to expedite prosecution of this Application, the Examiner is encouraged to contact the undersigned directly.

Respectfully submitted,

Date: 2/5/09

Kenneth R. DeRosa, Reg. No. 39,549
Stuart D. Rudoler, Reg. No. 45,059
Attorney for Applicant
Rudoler & DeRosa LLC
2 Bala Plaza, Suite 300
Bala Cynwyd, PA 19004
Tel: 610-660-7753
Fax: 267-940-3000